Project 2: Analyzing Data using Spark

Objectives

- Practice loading large data.
- Practice analyzing data using RDDs.
- Practice working with structured data in Spark.
- Running code on an AWS cluster.
- Working in groups and managing time.

Code

You will need to add your code to the provided project template available at: `https://github.itu.dk/imel/LSDA_S19_project2`

Datasets

You will be using the two datasets, follow these steps to download them:

- **GoodReads Best Books**.You can download it from `https://s3.amazonaws.com/goodreadreviews/book_data.csv`. This dataset is a cleaned version of the one available at: `https://www.kaggle.com/meetnaren/goodreads-best-books`.
- **Yelp**. You can download it from: `https://www.kaggle.com/yelp-dataset/yelp-dataset`.

Part I: Analyzing GoodReads Reviews

Input Data

In this part of the project, you analyze a dataset that includes the reviews for the best books on goodreads. The schema of the dataset is shown in Figure 1. You will need to download the data and store it in the path of "/data/goodreads-best-books" in your project directory. (Note: if you store it at a different location, you will need to change the path in application.conf)

Requirement

As shown in Figure 1, each review specifies a list of genres for a book. First, you are required to generate a list of all the distinct genres that appears for all reviews in the dataset. This means filling the code for the following function:

```
def findBookGenres(bookReviews : RDD[BookReview]) : List[String] = ???
```

```
book_authors: List[String],
book_desc : String,
book_edition: String,
book_format: String,
book_isbn:String,
book_pages: Int,
book_rating: Double,
book_rating_count: Int,
book_review_count: Int,
book_title:String,
genres:List[String],
image_url:String
```

Figure 1: Schema of Goodreads Dataset.

Second, you are required to count the books that are described by each genre appearing in the reviews and sort them in a descendant order using the following three approaches. The output is a list of pairs of (genre , count).

- Using a naive approach that processes a genre at a time and iterates through all the reviews of books counting those that are tagged by this genre. You will need to fill the code for the following functions:

```
def genreBookCount(genre: String,
                   bookReviews : RDD[BookReview]) : Int = ???
def rankingByCounting(bookReviewsRDD: RDD[BookReview],
                      bookGenres: List[String])
                                : List[(String,Int)] = ???
```

- You will first need to create an index of the genres, each pointing to the reviews that tags them. Fill in the code for this function:

```
def generateIndex(bookReviewsRDD: RDD[BookReview],
                  bookGenres: List[String])
                       : RDD[(String, Iterable[BookReview])] =  ???
```

  Next, you fill in the function that takes this index as an input and counts the reviews for each genre. The function signature is as follows:

```
def rankingUsingIndex(bookReviewsGenresIndex: RDD[(String,
                      Iterable[BookReview])])
                              : List[(String,Int)] = ???
```

Note that you will need to choose the right locations of your code to persist the RDDs otherwise the computation might fail.

Part II: Analyzing Yelp Reviews

Data

In this part of the project, you analyze a dataset that includes yelp reviews. The dataset is divided into several json files. We are interested in these three files: yelp_academic_dataset_review.json, yelp_academic_dataset_business.json, and yelp_academic_dataset_users.json. The schema of the datasets are shown at this link: `https://www.kaggle.com/yelp-dataset/yelp-dataset`. You will need to download the data and store it in the path of "/data/goodreads-best-books" in your project directory. (Note: if you store the files at a different location, you will need to change the path in application.conf)

Requirement

The dataset is loaded as Spark DataFrames. You are required to write the code to implement the following queries on the Spark DataFrames using both SQL statement and DataFrame transformations.

- **Q1:** Analyze yelp_academic_dataset_business.json to find the total number of reviews for all businesses. The output should be in the form of DataFrame with one value representing the count. You will need to write the code for these two functions:

```
def totalReviewsSQL(yelpBusinesses : DataFrame):DataFrame = ???

def totalReviewsbDF(yelpBusinesses : DataFrame):DataFrame = ???
```

- **Q2:** Analyze yelp_academic_dataset_business.json to find all businesses that have received 5 stars and that have been reviewed by 1000 or more users. The output should be in the form of DataFrame of (name, stars, review_count). You will need to write the code for these two functions:

```
def fiveStarBusinessesSQL(yelpBusinesses: DataFrame):DataFrame = ???

def fiveStarBusinessesDF(yelpBusinesses: DataFrame):DataFrame = ???
```

- **Q3:** Analyze yelp_academic_dataset_users.json to find the *influencer* users who have written more than 1000 reviews. The output should be in the form of DataFrame of user_id . You will need to write the code for these two functions:

```
def findInfluencerUserSQL(yelpUsers : DataFrame):DataFrame = ???

def findInfluencerUserDF(yelpUsers : DataFrame):DataFrame = ???
```

- **Q4:** Analyze yelp_academic_dataset_review.json, yelp_academic_dataset_business.json, and a view created from your answer to Q3 to find the businesses that have been reviewed by more than 5 *influencer* users. The output should be in the form of DataFrame of names of businesses that match the described criteria. You will need to write the code for these two functions:

```
def findFamousBusinessesSQL(yelpBusinesses: DataFrame,
       yelpReviews: DataFrame,
       influencerUsers: DataFrame) : DataFrame = ???
```

```
def findFamousBusinessesDF(yelpBusinesses: DataFrame,
       yelpReviews: DataFrame,
       influencerUsersDF: DataFrame): DataFrame = ???
```

- **Q5:** Analyze yelp_academic_dataset_review.json and yelp_academic_dataset_users.json to find a descendingly ordered list of users based on their the average star counts given by each of them in all the reviews that they have written. The output should be in the form of DataFrame of (user names and average stars). You will need to write the code for these two functions:

```
def findavgStarsByUserSQL(yelpReviews: DataFrame,
       yelpUsers: DataFrame):DataFrame = ???

def findavgStarsByUserDF(yelpReviews: DataFrame,
       yelpUsers: DataFrame):DataFrame = ???
```

Deliverable

You need to deliver the following:

- Your code.
- A report that includes the following:
  - A brief description of your solution.
  - A comparison of the performance of the three approaches implemented in part I.
  - A discussion of the type of optimization techniques employed by Spark for your queries in part II. (Hint: You will need to use *explain* or the Spark UI to show the execution plans)
  - A discussion of the effect on performance of using *persist* in your application.
  - An analysis showing the effect of increasing the cluster size.